Parallel Histogram Equalization Using CUDA

First-name Last-name

I. Introduction

This project implements a parallel histogram equalization algorithm for colour images using CUDA. Histogram equalization is a widely used technique to improve the contrast in images by redistributing the luminance values. In our solution, the image is first converted from RGB to YUV colour space so that the luminance (Y) component can be processed independently. We then compute the histogram and cumulative distribution function (CDF) of the Y channel, create a lookup table (LUT), and apply it to obtain the equalized image before converting back to RGB.

The implementation consists of a sequential CPU version, a baseline parallel CUDA version, and an optimized bonus version that uses shared memory for partial histogram computation. Our objective was to offload the entire algorithm to the GPU, minimize data transfers, and reduce atomic contention to achieve significant speed-ups.

II. Experiments

We evaluated the implementations on images of sizes: 720x480, 1024x768, 1920x1200, 3840x2160, and 7680x4320. Each experiment was repeated 7 times. The average execution times (in milliseconds) for each approach are summarized in Table I.

Image Size	Sequential	Parallel	Bonus
720x480	14.60 ms	0.74 ms	0.47 ms
1024x768	29.25 ms	1.00 ms	0.75 ms
1920x1200	80.46 ms	3.14 ms	1.61 ms
3840x2160	274.34 ms	7.75 ms	4.04 ms
7680x4320	989.65 ms	31.53 ms	15.23 ms

Table I
Average execution times for each implementation.

The speed-up factors, calculated as the ratio of sequential to parallel execution time, are shown in Table II. For larger images the baseline parallel version achieves a speed-up of up to 35× and the bonus version up to 68×.

Image Size	Parallel Speed-Up	Bonus Speed-Up
720x480	19.7×	31.0×
1024x768	29.3×	39.3×
1920x1200	25.6×	49.9×
3840x2160	35.4×	67.9×
7680x4320	31.4×	65.0×

Table II
Speed-up factors relative to the sequential implementation.

Figure 1 illustrates the speed-up achieved by both parallel versions. The plot clearly shows the increasing benefit of parallelization as image resolution grows.

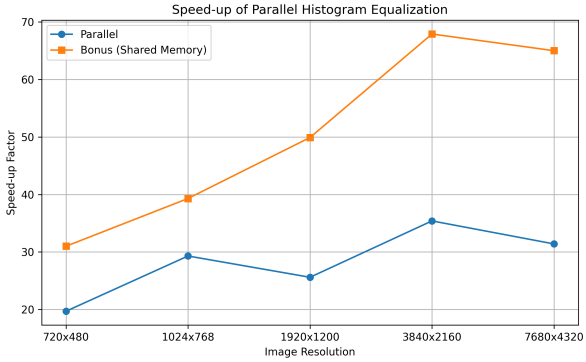

Figure 1. Speed-up comparison of the parallel and bonus implementations relative to the sequential version.

Discussion

The baseline parallel implementation shows significant improvement over the sequential version, largely due to the independent processing of pixels and effective use of CUDA atomic operations. However, the bonus version, which employs shared memory for computing partial histograms, reduces atomic contention and further improves performance, particularly for high-resolution images. Variations in sequential timings (e.g., an outlier in the 7680x4320 case) are likely due to system overhead and data transfer variability. Future work could further optimize kernel configurations and memory usage.

III. Conclusion

The parallel histogram equalization algorithm implemented using CUDA demonstrates substantial speed-ups compared to a sequential approach. The bonus version, leveraging shared memory optimizations, performs even better, especially on larger images. While the method works well for scientific imaging tasks, further improvements in atomic operation handling and memory synchronization may yield additional performance gains.